# Utilizando GPU en la Recuperación de Información Multimedia

**Mercedes Barrionuevo, Mariela Lopresti, Natalia Miranda, Fabiana Piccoli, Nora Reyes**
*LIDIC- Univ. Nacional de San Luis*
*San Luis, Argentina*
{mdbarrio, omlopres, ncmiran, mpiccoli}@unsl.edu.ar

## 1. Contexto

Esta propuesta de trabajo se lleva a cabo dentro del proyecto de investigación *"Nuevas Tecnologías para el Tratamiento Integral de Datos Multimedia"* abarcando las líneas de investigación "Computación de Alto Desempeño", "Procesamiento de Información Multimedia" y " Recuperación de Datos e Información Multimedia". Dicho proyecto se desarrolla en el marco del Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC), de la Facultad de Ciencias Físico, Matemáticas y Naturales de la Universidad Nacional de San Luis.

## 2. Resumen

Este trabajo presenta las distintas líneas de trabajo abordadas considerando la GPU como computadora masivamente paralela de propósito general. Los problemas a tratar están relacionados a las Bases de Datos Métricas, en este momento nos enfocamos en la optimización de la resolución de distintos tipos de consultas.
***Palabras Claves***: Computación de alta perfomance, Espacios métricos, Recuperación de Información.

## 3. Introducción

Una de las principales herramientas que las computadoras ofrecen es almacenar grandes cantidades de datos organizados, siguiendo un determinado esquema o un modelo de datos que facilite su almacenamiento, recuperación y modificación de una forma rápida y ordenada.

Cualquier conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso constituye una base de datos tradicional, en las cuales se busca sólo a partir de una clave. Las bases de datos y repositorios de información multimedia (audio, imagen, video y texto) no pueden ser trabajadas tan eficientemente como en las bases de datos tradicionales debido a que la información multimedia debe ser recuperada por similitud; mientras que en las bases de datos tradicionales esta búsqueda es exacta.

Las consultas que pueden ser de interés en bases de datos no estructuradas se denominan búsquedas por similitud o búsquedas por proximidad. La búsqueda por proximidad es la búsqueda en Bases de Datos de elementos que sean similares o cercanos a un elemento consultado. La similitud es modelada con la función de distancia, la cual satisface entre otras propiedades la desigualdad triangular. El conjunto de elementos u objetos sobre los que se define la función de distancia es llamado *espacio métrico* [3].

El costo computacional de resolver consultas en espacios métricos se mide en la cantidad de evaluaciones de funciones de distancia necesarias para responder a la consulta. Existen diferentes alternativas de optimización, una de ellas es el desarrollo de las soluciones computacionales aplicando técnicas de computación de alto desempeño para tecnologás emergentes como son las GPUs.

El poder computacional asociado a las GPUs (Unidad de Procesamiento Gráfico), el constante avance asociado a su hardware y software, y bajo costo, han constituido una alternativa válida a las supercomputadoras paralelas [1, 17]. Para cierto tipo de aplicaciones, una tarjeta de video puede proporcionar mucho más poder de cómputo que la computadora huésped [14]. Entre las características destacadas se encuentran la arquitectura de computación de alto rendimiento, el alto bandwith de memoria, los numerosos chips dedicados al procesamiento de datos y la administración de

las unidades de procesamiento.

Si consideramos el desarrollo de software para GPUs, no todo tipo de problemas puede ser resueltos en la GPU, los problemas más adecuados son aquellos que pueden ser implementados con procesamiento de stream y usando memoria limitada. Las aplicaciones más adecuadas son aquellas con cuantioso paralelismo de datos. El uso de la GPU para computaciones de propósito general se conoce como GPGPU (GPU de propósito general) [11, 19, 22].

La programación paralela de aplicaciones en la GPU debe considerar varias diferencias con la programación paralela en computadoras paralelas típicas, las más relevantes son: *El número de unidades de procesamiento*, *Estructura de la memoria CPU-GPU* y *Número de threads paralelos*. Respecto al número de unidades de procesamiento, a diferencia de las soluciones para computadoras masivamente paralelas donde el número de unidades de proceso coincide con el número de unidades de procesamiento (Procesadores o cores), en GPGPU esto no es considerado (Es posible la ejecución de cientos o miles de unidades de proceso en un número inferior de cores). En el caso de la estructura de la memoria del sistema CPU-GPU, existen dos espacios de memoria diferentes: la memoria del host (Compartida por todos los hilos de la CPU durante la aplicación) y la memoria de la GPU. Como los hilos o threads de la GPU ejecutan en un espacio de memoria separado a los threads del host en la aplicación, las transferencias de código y datos son necesarios entre el host. Finalmente, la programación de las GPU tiene la posibilidad de iniciar un gran número de threads con poca sobrecarga. Toda GPU tiene mecanismos transparentes y de bajo costo para la creación y administración de threads.

Existen diferentes alternativas para procesamiento de aplicaciones de propósito general en la GPU, la más ampliamente utilizada es la tarjeta Nvidia, para la cual se ha desarrollado un kit de programación en C, con un modelo de comunicación de datos y de control de hilos proporcionado por un driver, quien provee una interfaz GPU-CPU [10]. Este ambiente de desarrollo llamado Compute Unified Device Architecture (CUDA) propone un modelo de programación SIMD(Simple Instrucción, Múltiples Datos) con funcionalidades de procesamiento de vector [5].

Las líneas de investigación proponen utilizar la GPU como computadora masivamente paralela para resolver mediante técnicas de alto desempeño problemas relacionados a la adminsitración de base de datos métricas.

## 4. Líneas de Investigación y Desarrollo

Actualmente ha surgido la necesidad de crear bases de datos de información no estructurada, como por ejemplo imágenes, texto, sonido y video. Muchas aplicaciones computacionales necesitan buscar eficientemente información sobre estas bases de datos. Para realizar consultas en este tipo de bases de datos se han creado nuevos modelos y algoritmos de búsqueda más generales que los correspondientes a bases de datos tradicionales [3].

A estas nuevas bases de datos se las denomina base de batos métricas y las consultas se realizan según búsquedas por similitud o proximidad. La búsqueda por proximidad es la búsqueda en bases de datos de elementos similares o cercanos al elemento consultado. La similitud es modelada con una función de distancia, la cual satisface las siguientes propiedades. Si $X$ denota el universo de objetos válidos. Un subconjunto finito de él $U$, de tamaño $n$, es el conjunto de objetos o base de datos en donde se realizan las búsquedas. La función $d : X * X \longrightarrow \mathbb{R}^+$ denota la medida de distancia entre los objetos. Esta función de distancia debe cumplir las propiedades de positividad $\forall x, y \in X, d(x, y) \geq 0$, simetría $(\forall x, y \in X, d(x, y) = d(y, x))$, reflexividad $(\forall x \in X, d(x, x) = 0)$, desigualdad triangular $\forall x, y, z \in X, d(x, y) \leq d(x, y) + (d(y, z)$ y en la mayoría de los casos la positividad es estricta $(\forall x, y \in X, x \neq y \Rightarrow d(x, y) > 0)$. El conjunto de elementos u objetos sobre los que se define la función de distancia es llamado *espacio métrico* [3].

Mientras más pequeña sea la distancia entre los objetos más similares son dichos objetos. Existen tres tipos básicos de consultas por proximidad en espacios métricos:

- Consulta por rango$(q, r)_d$: recupera los elementos que están a lo más a distancia $r$ de $q$.
- Consulta por vecino más cercano $NN(q)$: recupera el vecino más cercano a $q$ en $U$.
- Consulta por $k$ vecinos más cercanos

$NN_k(q)$: recupera los $k$ vecinos más cercanos a $q$ en $U$.

Las líneas de investigación y desarrollo que actualmente se siguen están realacionadas a la resolución de consultas en espacios métricos. Para ello nos enfocamos en distintas áreas, ellas son:

- *Algoritmos de Ordenamiento*

  Cuando se trabaja con datos multimedia se utilizan Base de Datos métricas. En situaciones donde se quiere responder a consultas de estas bases de datos, no tiene sentido la búsqueda por exactitud, la idea es medir la similitud (o diferencia) entre el elemento de consulta dado y cada uno de los objetos de la base de datos. Para determinar la similaridad entre los objetos se deben obtener las distancias entre un elemento particular (consulta) y cada objeto de la base de datos. Existen diferentes funciones de distancia, una de ellas es la *Distancia de Edición*, la cual computa el número mínimo de operaciones requeridas para transformar un objeto en otro.

  Cuando se deben responder a cualquiera de las consultas planteadas en estas bases de datos es necesario, en la mayoría de los casos, ordenar resultados intermedios o finales. Dicha operación debe ser realizada en forma eficiente, por ello es de vital importancia lograr algoritmos con buen desempeño. Hemos considerado tres algoritmos de ordenamiento: *Quicksort*[9], *QuickSelect*[12] y *Parallel Sort Regular Sampling* (*PSRS*) [13], todos con diferentes características y conocidos por su buen desempeño en ámbitos secuenciales y paralelos.

  Existen bibliotecas estándares incluidas dentro de los repositorios de CUDA, las cuales implementan varios de estos métodos de ordenamiento, una de ellas es *Thrust* [8]. *Thrust* ofrece una colección de algoritmos paralelos tales como *scan*, *sort*, *reduction*, entre otros. En ella, los desarrolladores realizan su computación usando una colección de algoritmos de alto nivel y delegan completamente la decisión de cómo implementar la computación a la biblioteca. Esta interfaz abstracta le permite a los programadores describir qué computar sin colocar ninguna restricción adicional sobre cómo llevar a cabo la computación.

  Una desventaja de *Thrust* es la exposición de sólo un subconjunto de capacidades del hardware debido a las absracciones para los desarrolladores. Ante esto y el bajo desempeño obtenido en las consultas implementadas, se decidió buscar e implementar alternativas que lograran un mejor rendimiento. En ambas propuestas se tiene en cuenta la naturaleza paralela de la GPU y todas sus características para lograr óptimos resultados. Actualmente se ha desarrollado y evaluado *Quicksort* y *QuickSelect*, mientras que el *PSRS* está en las etapas finales de desarrollo.

- *Consultas en Espacios Métricos*

  En este punto estamos trabajando en dos direcciones, una se corresponde con la resolución de cada tipo de consulta aplicando el método de resolución de fuerza bruta y diferentes técnicas algorítmicas de computación de alto desempeño en la GPU. Esta línea se nutre de la línea anterior, donde es necesario contar con buenos algoritmos de ordenamiento para resolver las consultas.

  La otra línea de investigación dentro de esta área está la optimización a través del uso de índices.

  Si tenemos una base de datos de cardinalidad $n$, todas estas consultas pueden ser resueltas ejecutando $n$ evaluaciones de distancia. Generalmente el número de evaluaciones de distancias ejecutadas es la medida de complejidad de los algoritmos. El desafío es diseñar un algoritmo de indexación eficiente que reduzca el número de las evaluaciones de distancia [3, 4, 6, 7, 18, 20, 21, 23].

  Los algoritmos de indexación permiten construir a priori un índice, una estructura de datos capaz de ahorrar computaciones de distancias al responder consultas por proximidad.

  En general todos los algoritmos de indexación particionan el conjunto $X$ en subconjuntos $X_i$. Se construye un índice para permitir determinar una lista de subconjuntos $X_i$ de candidatos potenciales a contener objetos relevantes para la consulta.

Actualmente se está trabajando con el índice de *Permutantes*, el cual permite acelerar las consultas y responderlas por aproximación. La inclusión de técnicas de computación de alto desempeño nos permite optimizar aún más, no sólo el proceso de indexación sino también la obtención de las respuestas a las consultas.

En ambas líneas de investigación, las implementaciones han mostrado muy buenos resultados, algunos de los cuales han sido publicados en congresos nacionales e internacionales [15, 16].

## 5. Resultados obtenidos / esperados

En todas las líneas de investigación se han evaluado las soluciones en la GPU y comparado con otras existentes en la literatura. Los resultados obtenidos han mostrado una mejora sustancial en las velocidadas para resolver el problema. Además, las mejoras no sólo se reflejaron respecto a los parámetros de evaluación típicos de los sistemas de alto desempeño, sino también en la confiabilidad de las respuestas. Actualmente, se están desarrollando y evaluando otras soluciones y métodos de indexación.

## 6. Formación de Recursos Humanos

Los resultados esperados respecto a la formación de recursos humanos son hasta el momento una tesis de maestría y dos tesis de doctorado en desarrollo y otra en definición.

Además la finalización de beca de postgrado tipo II otorgada por el Consejo Nacional de Investigaciones Científicas (CONICET), obtenida el 01/04/11.

## Referencias

[1] Buck, I. - GPU computing with NVIDIA CUDA - SIGGRAPH '07 - ACM SIGGRAPH 2007 courses ACM - New York, NY, USA. 2007.

[2] Bustos Cárdenas, B.E. - Index Structures for Similarity Search in Multimedia Databases - PhD thesis, Universitat Konstanz, Fachbereich Informatik, Germany - Octubre 2006.

[3] Chávez E., Navarro G., Baeza Yates R.A., Marroquín J.L. - Searching in metric spaces - ACM Comput. Surv. Vol 33 N3 - Pp 273:321 - 2001.

[4] Chávez, E. and Navarro, G. - A compact space decomposition for effective metric indexing. - Pattern Recognition Letters 26(9) - Pp 1363:1376 - 2005.

[5] NVIDIA - NVIDIA CUDA Compute Unified Device Architecture, Programming Guide Version 5 - NVIDIA. 2013.

[6] Dohnal, V., Gennaro, C., Savino, C. and Zezula, P. - D-index: Distance searching index for metric data sets. - Multimedia Tools and Applications (MTAP), - 21(1): 9:33 - 2003.

[7] Figueroa Mora, K. - Indexación Efectiva de Espacios Métricos usando Permutaciones. - PhD thesis - Universidad de Chile, Santiago, Chile - 2007. - Director: Dr. G. Navarro y Dr. E. Chávez.

[8] Hoberock, J. Bell, N. - Thrust: A Parallel Template Library - http://www.meganewtons.com/. 2010.

[9] Hoare, C. - Quicksort - The Computer Journal. Vol 5, N. 1. Pp 10-16. 1962.

[10] Joselli, M., Zamith, M., Clua, E., Montenegro, A., Conci, A., Leal-Toledo, R. Valente, L., Feijo, B., Dórnellas, M., Pozzer, C - Automatic Dynamic Task Distribution between CPU and GPU for Real-Time Systems - 11th IEEE International Conference on Computational Science and Engineering, 2008 ( CSE '08) - Pp 48:55 - July 2008.

[11] Kirk D. and Hwu, W. - Programming Massively Parallel Processors, A Hands on Approach - Elsevier, Morgan Kaufmann. ISBN 978-0-12-381472-2. 2010.

[12] Kuba, M. - On Quickselect, partial sorting and Multiple Quickselect - Inf. Process. Lett. Vol 99 N. 5. Pp 181-186. 2006.

[13] Li, X., Lu, P., Schaeffer, J., Shillington, J., Wong, P., Shi, H. - On the Versatility of Parallel Sorting by Regular Sampling- Parallel Computing. Vol 19. Pp 1079-1103. 1993.

[14] Lloyd, D., Boyd, C., Govindaraju, N. - Fast computation of general Fourier Transforms on GPUS - IEEE International Conference on Multimedia and Expo - Pp 5:8 - April 2008.

[15] Lopresti, M., Miranda, N., Piccoli, F., Reyes, N. - Efficient Similarity Search on Multimedia Databases XVIII Congreso Argentino de Ciencias de la Computación, CACIC 2012. Pp 1079-1088. Bahía Blanca, Argentina. 2012.

[16] Lopresti, M., Miranda, N., Piccoli, F., Reyes, N. - Solving Multiple Queries through the Permutation Index in GPU. 4th International supercomputing Conference in Mexico. Colima - México. 5-8 March 2013. In Press.

[17] Luebke, D., Humphreys, G. - How GPUs Work Computer - vol 40, N 2 - Pp 96:100 - ISSN 0018-9162 - Feb. 2007.

[18] Mamede, M. - Recursive lists of clusters: A dynamic data structure for range queries in metric spaces. In Proc. 20th Intl. Symp. on Computer and Information Sciences (ISCIS'05), LNCS 3733 - pages 843:853 - 2005.

[19] Piccoli, M.F. - Computación de alto desempeño en GPU. ISBN: 978-950-34-0759-2. Edlup. Octubre 2011.

[20] Reyes, N. - Indices dinámicos para espacios métricos de alta dimensionalidad. - Master's thesis - Universidad Nacional de San Luis - San Luis, Argentina - 2002. - Director: Dr. G. Navarro.

[21] Samet, H. - Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). - Morgan Kaufmann Publishers Inc. - San Francisco, CA, USA - 2005.

[22] Sanders, J. and Kandrot, E. - CUDA by Example, An Introduction to General Purpose GPU Programming - Addison Wesley. ISBN 978-0-13-138768-3. 2010.

[23] Zezula, P., Amato, G., Dohnal, V. and Batko, M. - Similarity Search: The Metric Space Approach (Advances in Database Systems). - Springer-Verlag New York, Inc. - Secaucus, NJ, USA - 2005. - XVIII, 220 p., Hardcover - ISBN: 0-387-29146-6.